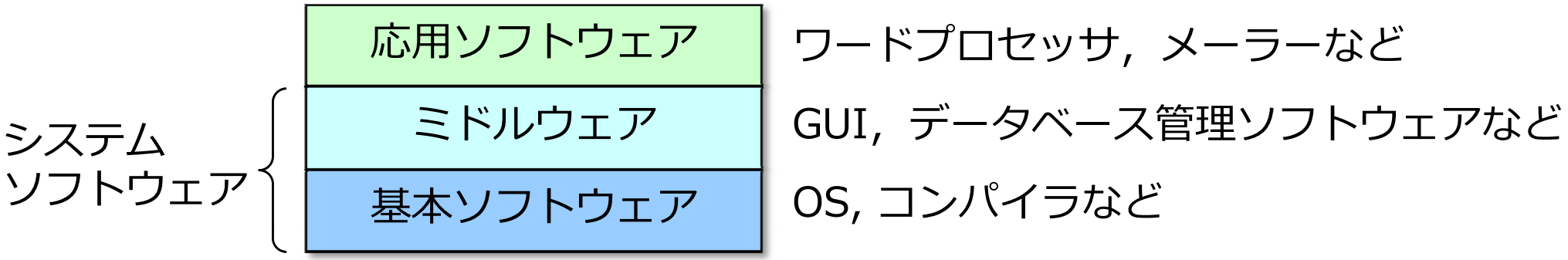
ソフトウェア工学
(part 1)

情報メディアシステム分野
手塚太郎

この資料は以下の場所にあります。
http://xi.kc.tsukuba.ac.jp/lec/SE1.pdf

ソフトウェア(Software)

- データ処理システムを機能させるための，プログラム，手順，規制，関連文書などを含む知的な創作
  - プログラム
  - 要求定義書，外部設計書，内部設計書，データベース定義書，コーディング規約，取扱説明書，運用マニュアル
  - プログラム，プログラムを作成する過程で得られるシステム設計書，フローチャートをはじめとする設計書，および，プログラム説明書などの関連資料
- cf. ハードウェア(hardware)
  - コンピュータ装置



ソフトウェアとハードウェアの違い

- ソフトウェア
  - 経年劣化なし
  - 導入後に修正可能
  - 製品の量産コスト，配布・流通コストは低い

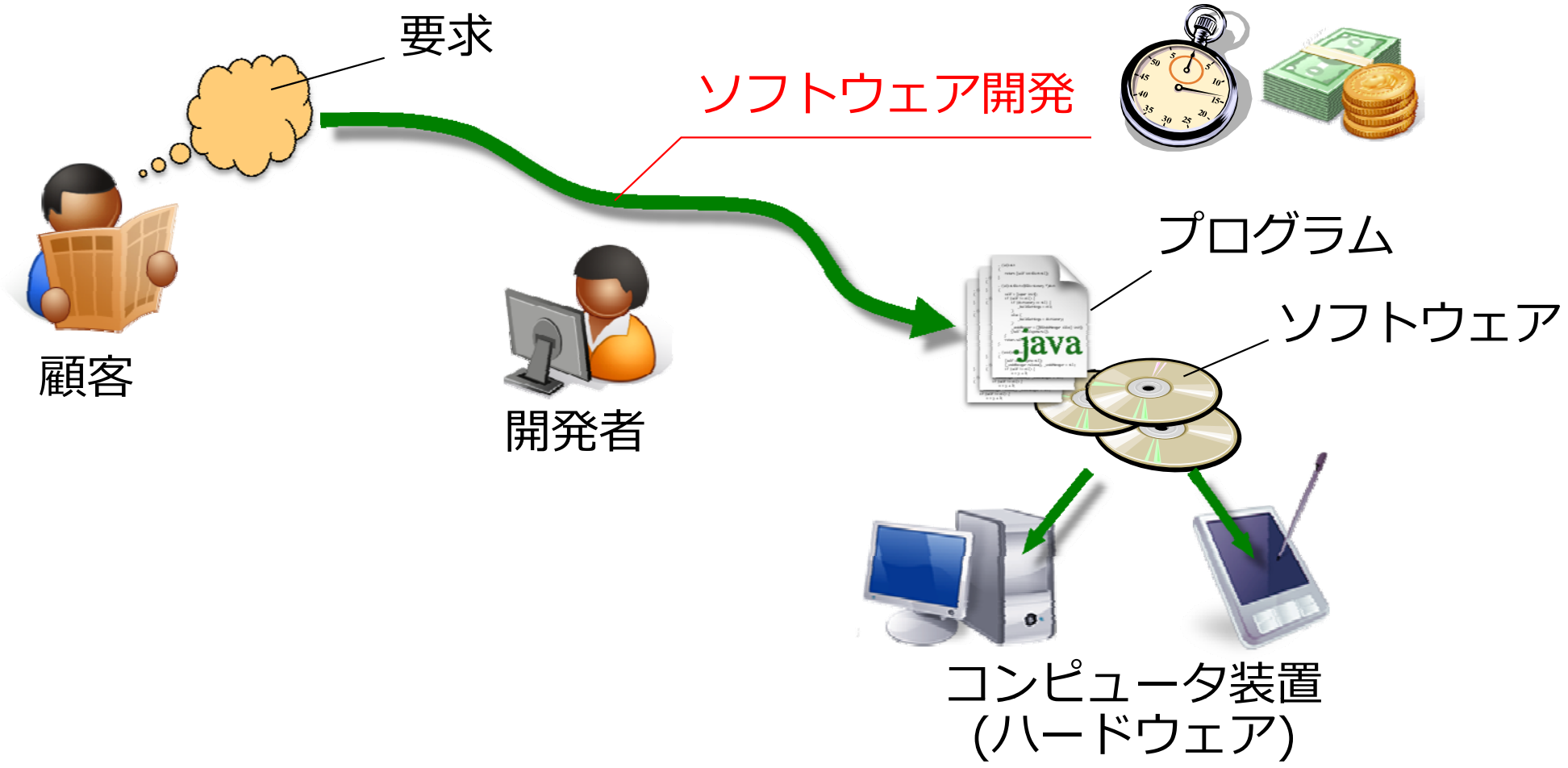
機能拡張，性能改善，環境適合に関する要求
ソフトウェア進化が前提

- ハードウェア
  - 経年変化あり (磨耗，部品の寿命)
  - 導入後の修正はほぼ不可能
  - 製品の量産および配布コストあり

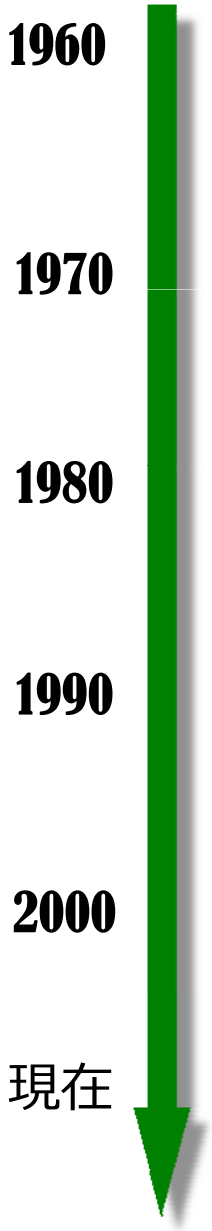
機能や性能を維持することに対する要求

ソフトウェア開発(Software Development)

- 顧客の要求をソフトウェア製品に変換する作業
  - 良いソフトウェアを効率的に構築
  - 品質(良さ)とコスト(時間/お金)のバランスが重要



ソフトウェア開発技術の歴史



1960

機能の時代:
既存業務の情報システム化

IBM OS/360 (1964): 500万行(推測)

1970

スケジュールの時代:
開発計画やライフサイクルの登場

1980

コストの時代:
開発における生産性が重要視

金融システム(1980半ば): 500万行

1990

品質の時代:
社会の依存度が増大
使い勝手の良さに対する要求

DVDレコーダ(2000): 20万行

自動車(2000): 100万行

携帯電話(2001): 100万行

2000

現在

複雑さへの挑戦

金融システム: 6400万行
DVDレコーダ(2000): 100万行
自動車: 500～1000万行
携帯電話: 500万行

WindowsXP(2002):
4000～7000万行(推測)

ソフトウェア工学(Software Engineering)

- コンピュータソフトウェアを対象として，その構築，運用，保守における生産性と品質の向上を実現するための技術体系や学問体系
  - 方法論(methodology)
  - 技法(technique) / 道具(tool)
  - プロジェクト管理(project management)
- ソフトウェア危機(software crisis)を打開するためのソフトウェア開発の技術体系および学問体系
- ソフトウェア工学の目的
  - 良いソフトウェアを開発すること
    - ソフトウェアの品質特性 (国際規格 ISO 9126)
- ソフトウェア工学の実践
  - ソフトウェアを開発するための理論，原理，技術に基づく方法論や，表記法，ツールを活用して，ソフトウェアを構築，運用，保守する活動
  - 大規模・高信頼ソフトウェアの開発 ≠ プログラミング

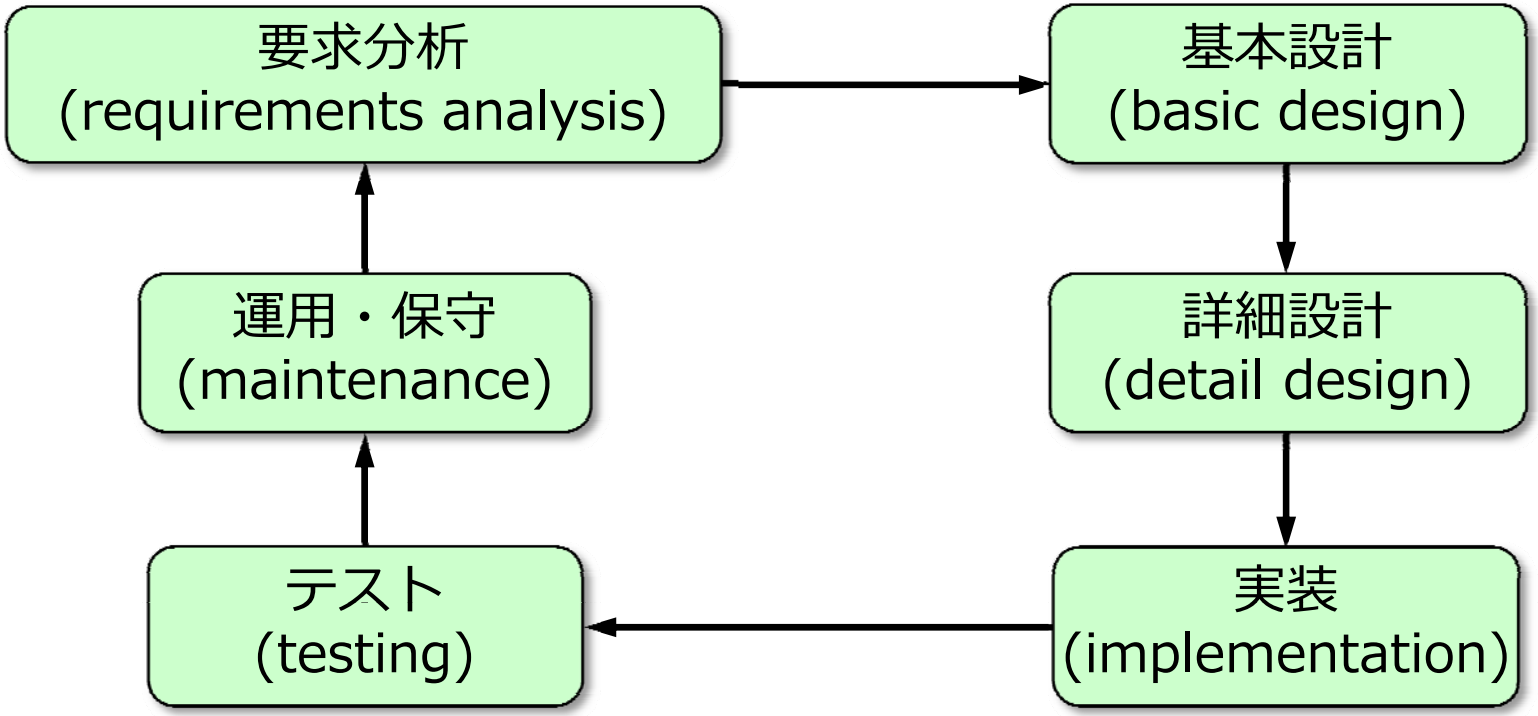
ソフトウェア危機

- 技術者不足，納期遅れ，品質低下，開発費増大
  - NATO会議(1968年)で指摘
- 生産物の側面
  - ハードウェアの大型化に伴うソフトウェアの規模の拡大
  - コンピュータの普及に伴う開発ソフトウェアの数の増大
- 人の側面
  - ステークホルダー(stakeholder)の多様化
    - ソフトウェア開発とのその成果物に対する利害関係者
- 時間の側面
  - 開発時間および利用時間の増大
  - 動作環境や社会的要求が変動
- 役割の側面
  - コンピュータで扱う分野の拡大
  - 社会的に重要なコンピュータシステムに対する信頼性の向上
  - ソフトウェアの大衆化に伴う使い勝手の向上

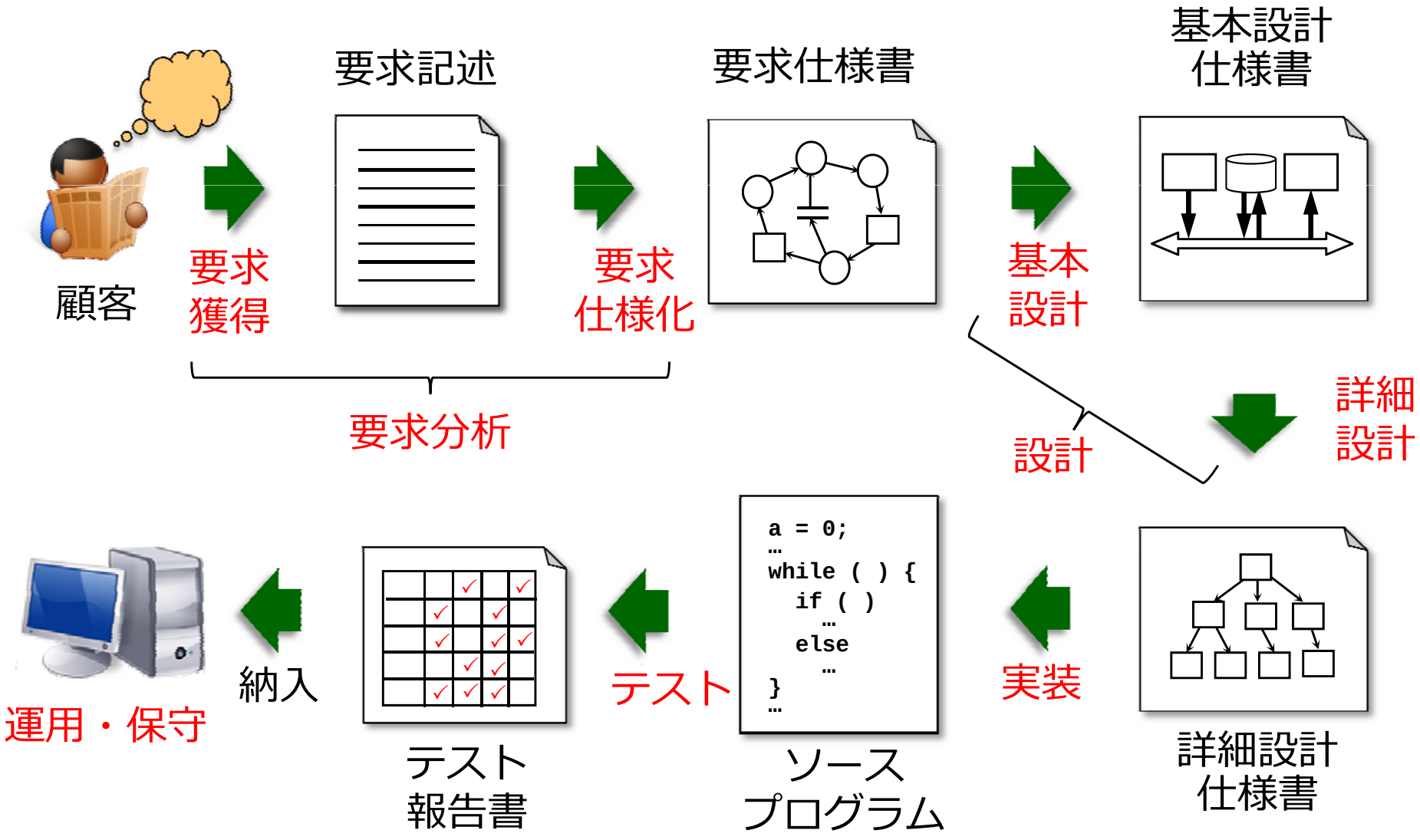
ソフトウェアの開発工程

ソフトウェア開発工程とは

- ソフトウェアの開発をどのように進めるかを規定したもの
  - 開発における作業: 開発プロセス(process)
  - 各作業において，プロダクト(product)を生成
    - 文書，図表，プログラム，マニュアルなど
- ソフトウェアプロセスモデル(process model)，ライフサイクル(life cycle)ともいう



ソフトウェア開発プロセス



ソフトウェア開発プロセス(要求分析)

- 要求獲得
  - 利用者(ユーザ) が要求するシステムを整理し，自然言語で文書化
    - 要求記述を作成

- 要求定義
  - どのようなシステムを作成するのかを決定
  - システム要求書を分析し，開発するシステムを形式的に文書化
    - 要求仕様書(requirements specification)
  - 分析者(analyst)が実施

ソフトウェア開発プロセス(設計)

- 基本設計
  - システム設計(system design)，外部設計(external design)ともいう
  - 外部から見たシステム全体をどのように作成するのかを決定
  - ソフトウェアアーキテクチャやユーザインタフェースを決定
    - 基本設計仕様書
  - 設計者(designer)が実施

- 詳細設計
  - プログラム設計(program design)，内部設計(internal design)ともいう
  - 個々のモジュールのアルゴリズムやデータ構造を決定
    - 詳細設計仕様書
  - プログラマ(programmer)が実施

ソフトウェア開発プロセス(実装，テスト)

- 実装
  - コーディング(coding)ともいう
  - プログラム設計仕様をプログラムに変換
  - 具体的なプログラミング言語(program language)による記述
    - ソースプログラム(source program)
  - プログラマが実施

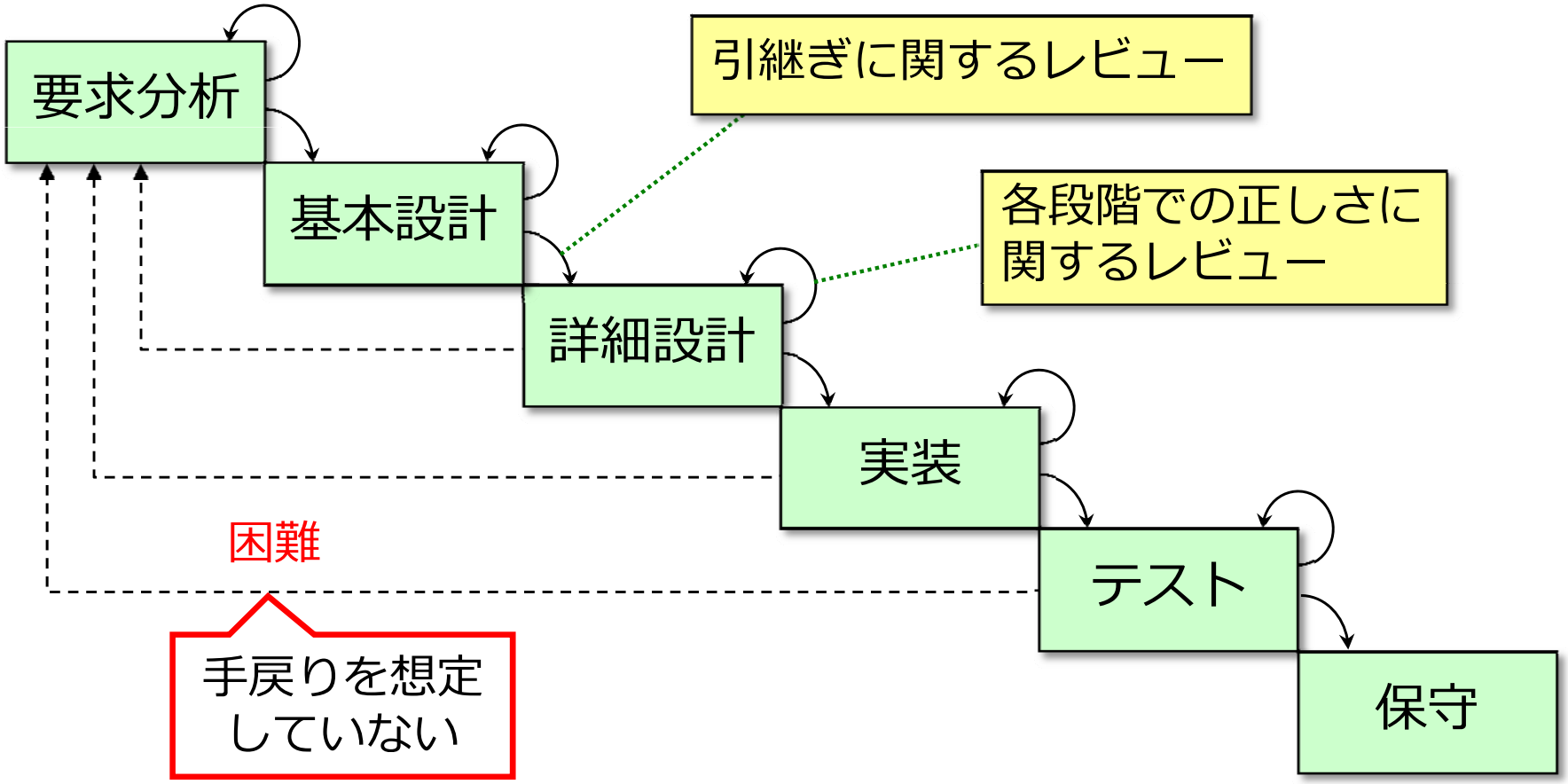
- テスト
  - 仕様書どおりにプログラムが動作するかどうかを検査
    - テスト報告書(test report)
  - 試験者(tester)が実施
    - ソフトウェア作成と独立の組織を用意
  - デバック(debug)
    - エラー(error)の修正は設計者やプログラマが実施)

ソフトウェア開発プロセス(運用・保守)

- 運用・保守
  - 納入後のソフトウェアを維持・管理
  - 運用段階で検出された故障(残存エラー)の修正
  - 変更要求への対応
    - 新機能の追加
    - 既存機能の変更
    - 新しい環境への適合
  - 保守者(customer engineer)が実施

ウォーターフォールモデル

- トップダウンな開発プロセス: 滝(waterfall)

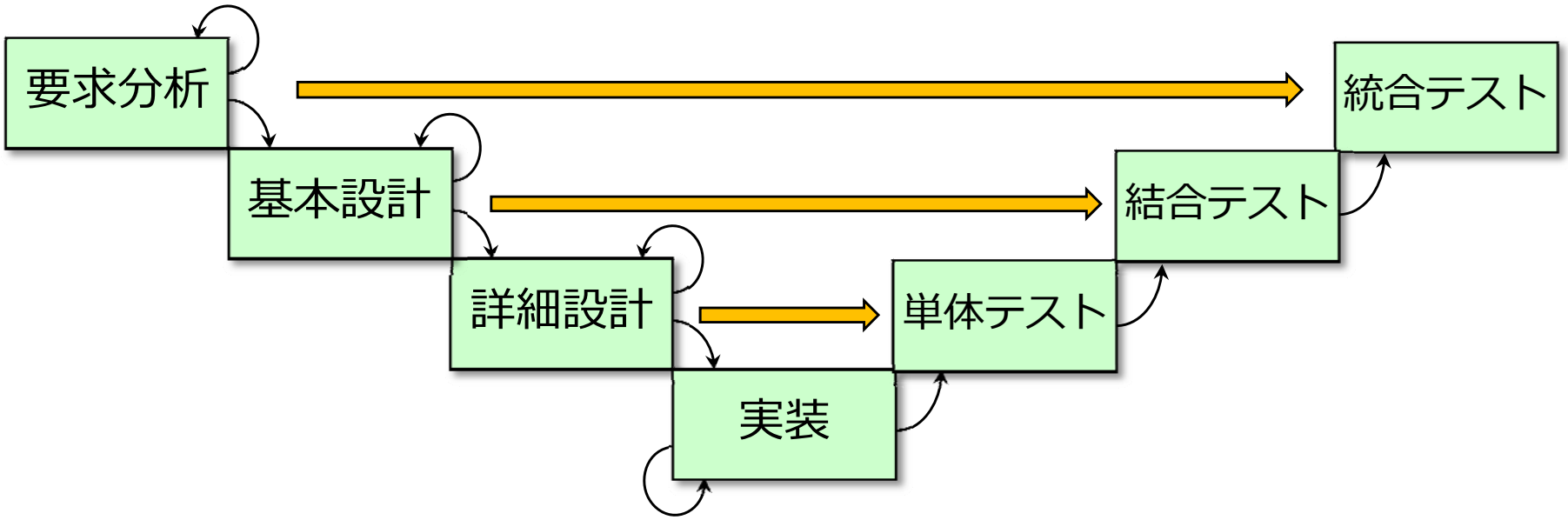


利点: 工数見積もりや進捗管理が容易

欠点: 仕様変更に弱い，誤り発見の遅れ，修正コストの増大

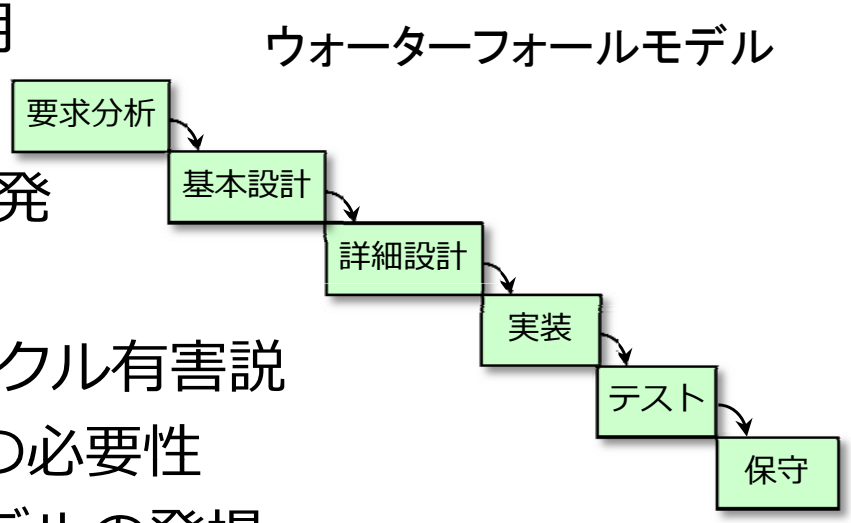
V字モデル

- ウォーターフォールモデルはV字モデルとして捉え直せる。



ソフトウェア開発モデルの推移

- 1960年代: 流れ図(flowchart)を利用
  - 開発方法論なし
- 1970年代: 構造的なソフトウェア開発
  - ウォータフォールモデル
- 1980年代: ソフトウェアライフサイクル有害説
  - 新しいソフトウェア開発モデルの必要性
- 進化型(成長型，発展型)プロセスモデルの登場
  1) 開発の初期段階から実行可能なプログラムを部分的に作成
  2) プログラムを実際に動作させて機能を確認
  3) プログラムの改善
  4) (2)と(3)を繰り返し
  - 使い捨て型プロトタイピング(throwaway prototyping)
  - スパイラルモデル(spiral model)
  - 進化型プロトタイピング(evolutionary prototyping)
  - アジャイルプロセスモデル(agile process model)



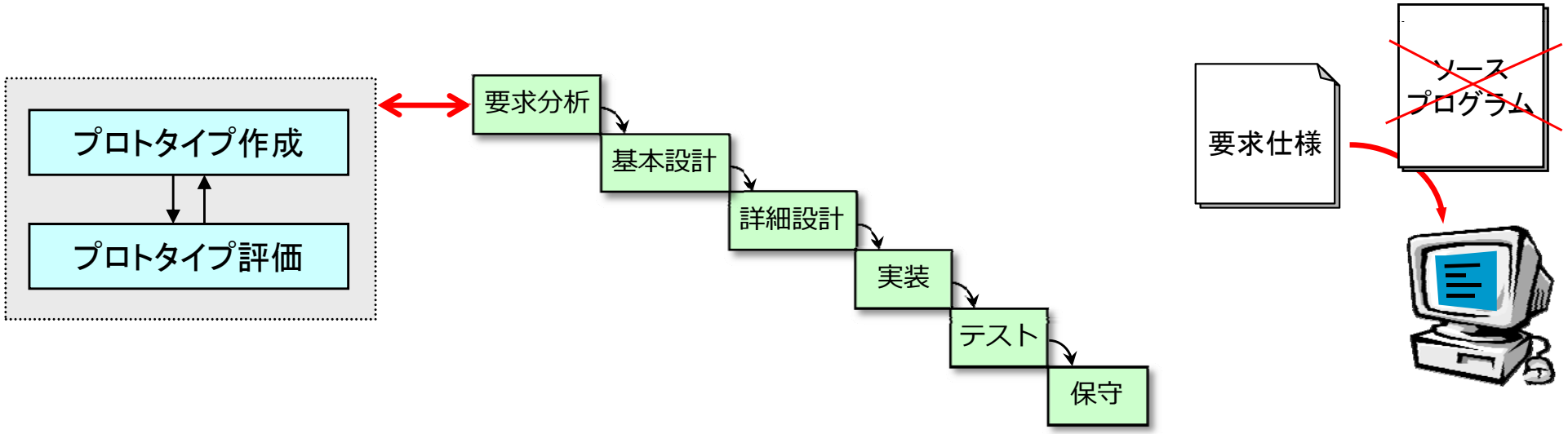
新しいソフトウェア開発モデル

1) ソフトウェアプロトタイピング(software prototyping)

- ✓ システム設計時に試作品(プロトタイプ; prototype)を構築

2) 操作的アプローチ(operational approach)

- ✓ 実行可能な仕様(executable specification)の作成



新しいソフトウェア開発モデル(cont'd.)

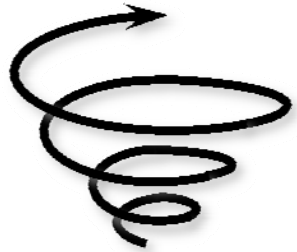
3) オブジェクト指向ソフトウェア開発(object-oriented software development)

- ✓ オブジェクトを単位としてソフトウェアを構築

  オブジェクト: 実世界の「もの」や「役割」などを抽象化したもの

- ✓ スパイラルモデル(spiral model)

  インクリメンタル(incremental) + イテラティブ(iterative)



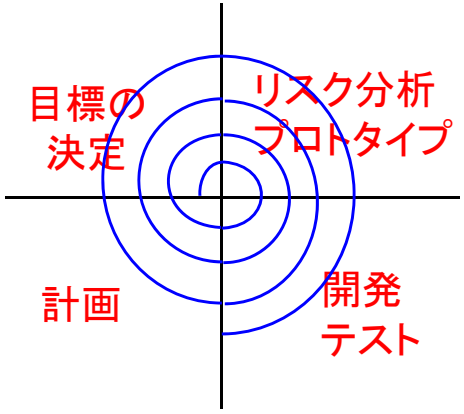
4) アジャイルソフトウェア開発(Agile software development)

例) XP(extreme programming)

アジャイルアライアンス宣言(manifesto)

- ✓ プロセスやツールよりも，個人や人同士の交流を重視
- ✓ 包括的なドキュメントよりも，動作するソフトウェアを重視
- ✓ 契約上の交渉よりも，顧客との協調を重視
- ✓ 計画に従うことよりも，変化に対応することを重視

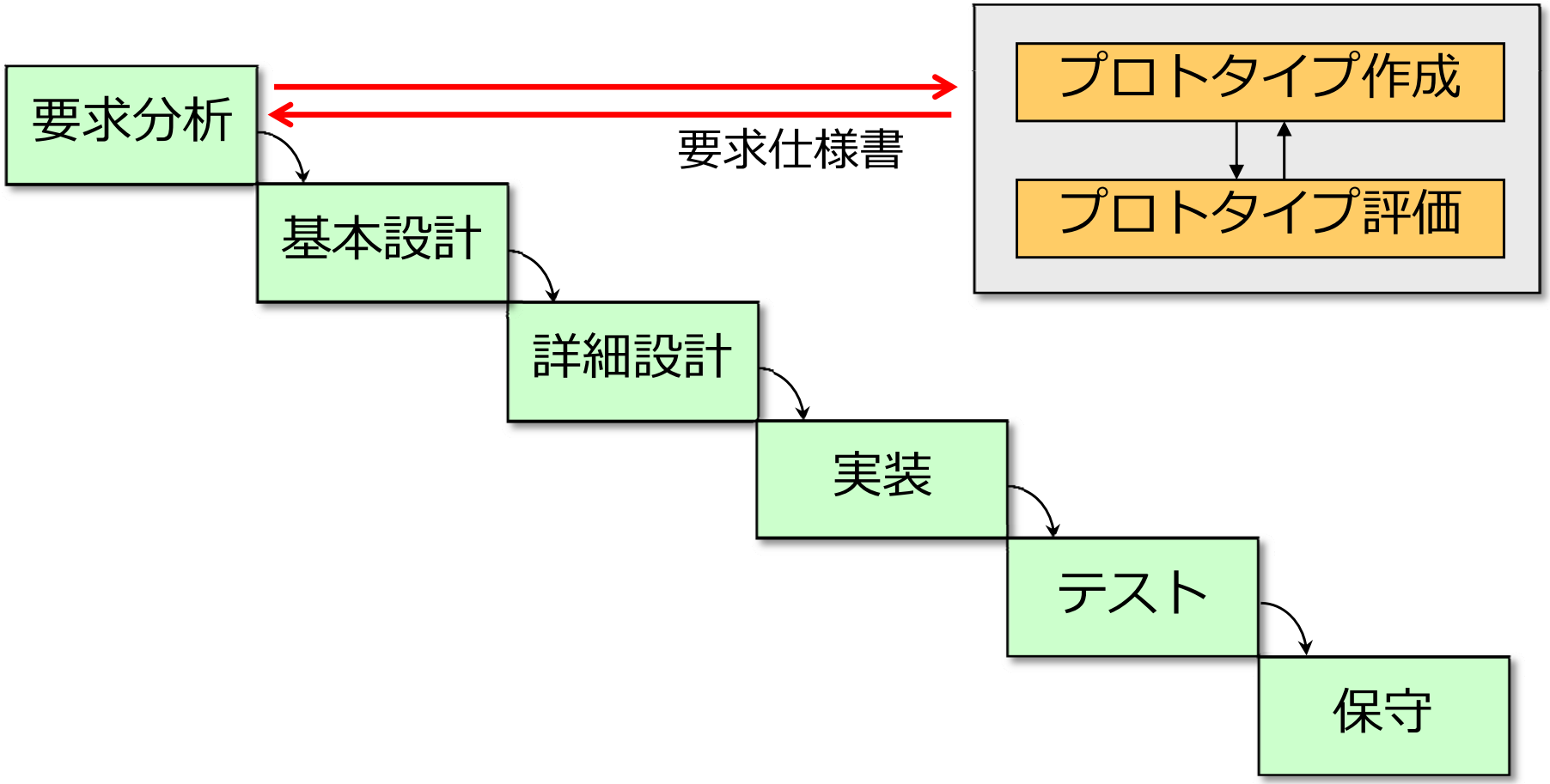
  注) 左側の項目を軽視するという意味ではない



Boehm (1988)

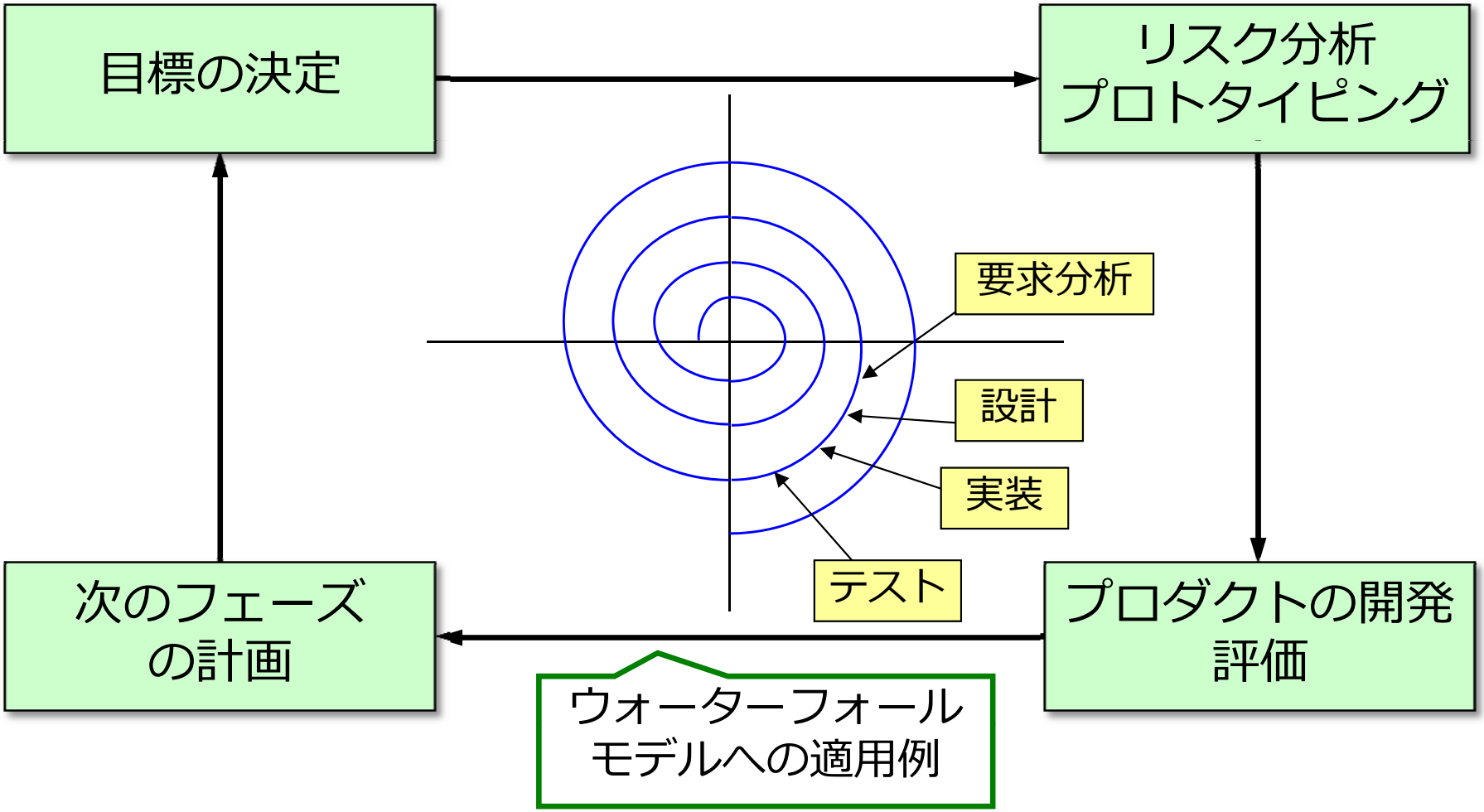
使い捨てプロトタイピング

- システム設計時にプロトタイプ(試供品: prototype)を構築
  - 問題点の早期発見と要求のあいまいさの解消
    - プロトタイプに基づき要求仕様書を作成
  - 大幅な手戻りを減少



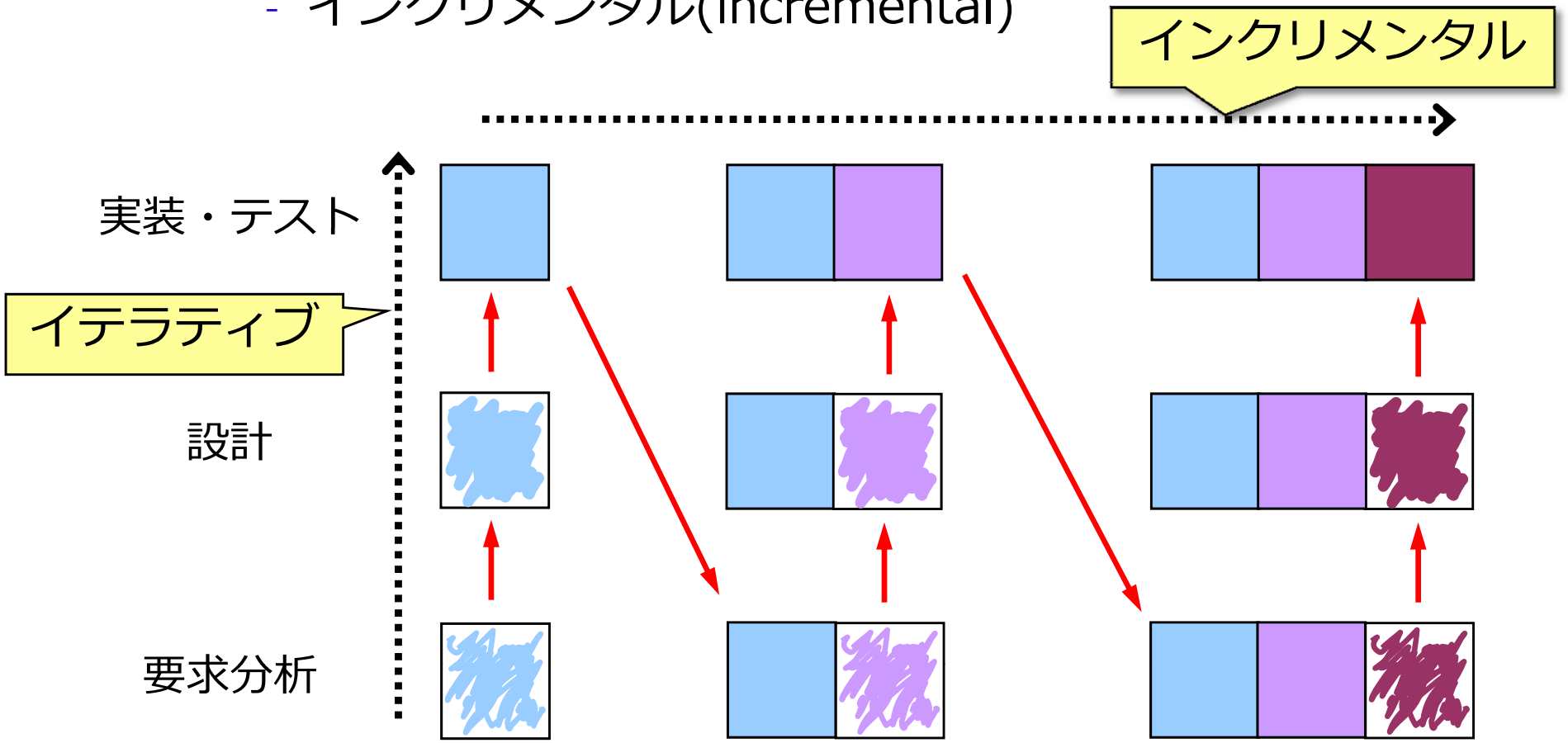
スパイラルモデル

- プロトタイプを作成し，利用者からのフィードバックに対応しながら，徐々にシステムを作成
  - リスク(開発の失敗)を分析することで，リスクを軽減



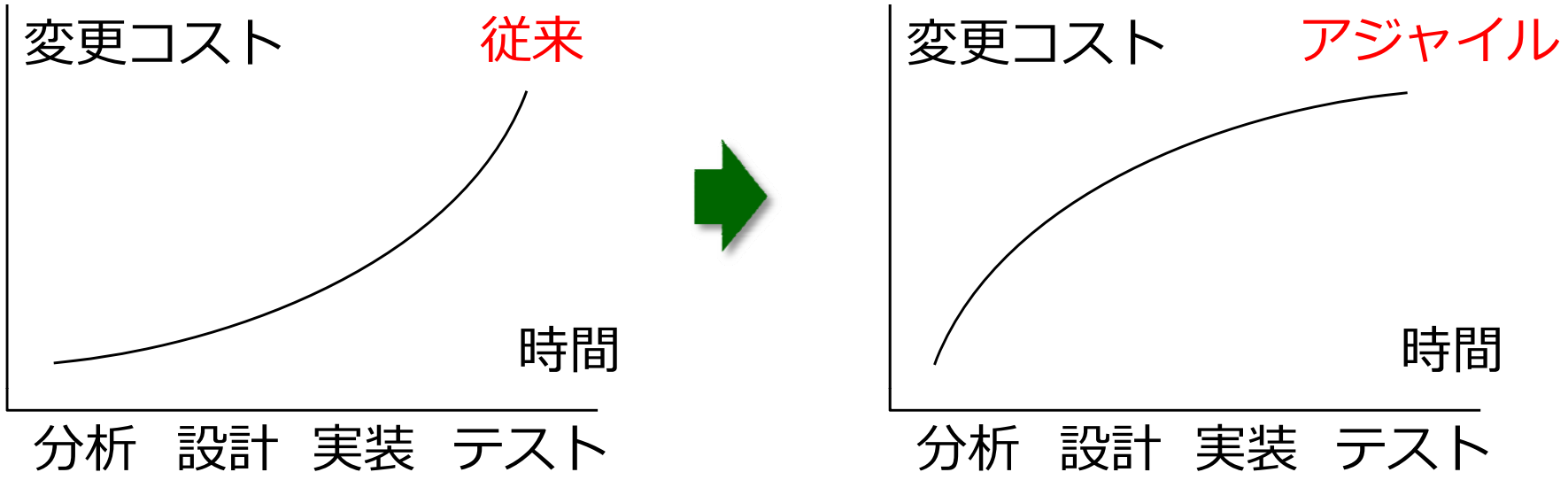
進化型プロトタイピング

- プロトタイプを徐々に修正し，そのまま完成品として利用
  - 機能が明確な部分から構築
  - 反復型開発モデル
    - イテラティブ(iterative)
    - インクリメンタル(incremental)



アジャイルプロセスモデル

- 変化に迅速に対応
  - 軽量(lightweight)プロセス
    - XP(extreme programming), Scrumなど
- 開発対象を多数の小さな機能に分割して，各機能を短期間で開発
  - 1つの反復(iteration)で1機能を実現
  - 実行可能なソフトウェアを随時提供
  - 究極の反復型開発(進化型プロトタイピング)



XP（エクストリームプログラミング）

- アジャイル開発の代表例のひとつ
- 開発時における具体的なプラクティス（行動指針）の集まり

当初のXPにおける12のプラクティス（前半）

計画ゲーム The planning game	次回リリースの範囲を早急に決める
小規模リリース Small releases	リリースサイクルを短めに
比喩 Metaphor	システムの動きを表すメタファー（比喩）を導入
シンプルデザイン Simple design	余分な複雑さを減らす
テスティング Testing	プログラマが継続的にテストを書く
リファクタリング Refactoring	コードの見直しを積極的に行う

XP（エクストリームプログラミング）

当初のXPにおける12のプラクティス（後半）

ペアプログラミング Pair programming	2人のプログラマが組になってコードを書く
共同所有権 Collective ownership	すべてのプログラマがすべてのコードにアクセス可能
継続的インテグレーション Continuous integration	常にテストを実行し、動作する状態におく
週40時間 40 hour week	働き過ぎは良い結果を生まない。
オンサイト顧客 On site customer	ユーザをチームに加える
コーディング標準 Coding standards	全プログラマがコーディング標準に従う

- 現在では19のプラクティスまで増えている。

ソフトウェアライフサイクルプロセス

SLCP（ソフトウェアライフサイクルプロセス）

- ソフトウェア開発とその周辺の業務手順における**作業項目を定義**し、**標準化**したもの。
- 他社に開発依頼する場合、作業手順に関する用語や定義が異なると、トラブルが生じうるため、統一が必要。
- 「ソフトウェア開発モデル」と異なり、開発の周辺業務も含まれている。
- プロジェクト管理において、管理の対象となる。

SLCPの要素

プロセス群	プロセス
主ライフサイクルプロセス	取得、供給、契約の変更管理、企画、要件定義、開発、運用、保守
支援ライフサイクルプロセス	文書化、構成管理、品質保証、検証、妥当性確認、共同レビュー、監査、問題解決、ユーザビリティ（使用性向上）
組織に関するライフサイクルプロセス	管理、環境整備、改善、人的資源、資産管理、再利用プログラム、ドメイン技術
システム監査プロセス	

開発プロセス

1. プロセス開始の準備
2. システム要件定義
3. システム方式設計
4. ソフトウェア要件定義
5. ソフトウェア方針設計
6. ソフトウェア詳細設計
7. ソフトウェアコード作成およびテスト
8. ソフトウェア結合
9. ソフトウェア的確性確認テスト
10. システム結合
11. システム適合性確認テスト
12. ソフトウェア導入
13. ソフトウェア受け入れ支援

運用プロセス

1. プロセス開始の準備
2. 運用テスト
3. 業務およびシステムの移行
4. システム運用
5. 利用者教育
6. 業務運用と利用者支援
7. システム運用の評価
8. 業務運用の評価
9. 投資効果および業務効果の評価

保守プロセス

1. プロセス開始の準備
2. 問題把握および修正分析
3. 修正の実施
4. 保守レビューおよび受け入れ
5. 移行
6. ソフトウェア廃棄